

Sommatore a 3 input (v1)

Stefano Scarcelli & Michele De Fusco

01 Dic 2023

Contents

1	Analisi progettuale	3
1.1	Analisi preliminare	3
1.2	Struttura del progetto	3
2	Implementazione	3
2.1	Adder n-bit	3
2.1.1	Codice VHDL	4
2.1.2	Sintesi	4
2.2	Register n-bit	4
2.2.1	Codice VHDL	4
2.2.2	Sintesi	4
2.3	Synched adder	4
2.3.1	Codice VHDL	4
2.3.2	Sintesi	4
2.4	Three adder (main)	4
2.4.1	Codice VHDL	4
2.4.2	Sintesi	4
3	Testing	4

1 Analisi progettuale

1.1 Analisi preliminare

L'obiettivo è quello di costruire un circuito in grado di sommare 3 numeri a n -bit (*2's complements*) e restituirne il risultato. Sia gli input che gli output devono essere sincronizzati tramite l'uso di registri.

L'idea di base è quella di usare due **Ripple carry** *parametrici* in cascata tra di loro per eseguire il calcolo desiderato $A + B + C = R$.

1.2 Struttura del progetto

L'idea alla base dell'implementazione è quella di inserire in pipeline i due moduli **sommatori** per eseguire prima la somma $A + B$ e successivamente $(A + B) + C$.

Questa implementazione porta all'inserimento di (in totale) di 6 **registri**, 2 in ingresso ad ogni adder, uno in aggiunta all'input C per salvare il risultato per la seconda operazione di somma e uno in uscita per salvare il risultato dell'operazione complessiva (richiesto dalle specifiche del progetto).

Il primo ritardo per ricevere un risultato coerente è di 2 colpi di clock mentre il delay per ricevere i risultati successivi al primo è di solo 1 colpo di clock.

2 Implementazione

L'implementazione si basa sulla definizione di una struttura gerarchica di componenti, partendo dalla definizione comportamentale dei componenti elementari (**Adder n-bit** e **Register n-bit**) per poi andare a comporre (tramite 2 livelli di astrazione, **Synched adder** e in fine l'elemento principale **Three adder (main)**) la struttura del progetto.

2.1 Adder n-bit

L'implementazione dell'**Adder n-bit** segue la descrizione comportamentale tramite **Ripple carry** parametrico usando i segnali *propagate* (P) e *generate* (G).

2.1.1 Codice VHDL

2.1.2 Sintesi

2.2 Register n-bit

L'implementazione dell'**Register n-bit** segue la descrizione comportamentale classica con memorizzazione a *fronti di salita*.

Il **registro** implementa in più un segnale di *clear* (asincrono) attivo alto.

2.2.1 Codice VHDL

2.2.2 Sintesi

2.3 Synched adder

Questo è un componente intermedio che abbiamo impostato per poi costruire il circuito completo.

Consiste in un **Adder n-bit** con 2 **registri** collocati agli input di esso.

2.3.1 Codice VHDL

2.3.2 Sintesi

2.4 Three adder (main)

Il circuito finale comprende invece l'uso di 2 **Synched adder** e 2 **registri** aggiuntivi, uno collocato tra l'input *C* del circuito e il secondo ingresso del secondo **Synched adder** (usato come buffer per salvare il valore di *C* nella pipeline) e un'altro tra il risultato dell'operazione e l'output del circuito (come richiesto dalle specifiche del progetto).

2.4.1 Codice VHDL

2.4.2 Sintesi

3 Testing